Network Geography

Srini, rs@toprllc.com

September 6, 2020

1 Introduction

Geographic Information Systems have revolutionized personal travel with **GPS** services and accessible maps. In this project we explore the nature of geographic data and how to manipulate it. At the core of this is the coordinate system that incorporates a Latitude and a Longitude at a location. Depending on the GPS source, the accuracy of this varies. In addition, the coordinates at a location gives us a potential way to calculate the distance to another location.

In this projectlet, we try to **discover** some public IP Addresses and ascertain their coordinates. In addition, estimates of distances between pairs of IP addresses is computed.

A totally artificial example with no practical utility is chosen strictly for illustration.

1.1 Learning Objectives

- Explore some geospatial information. Given an IP address, we attempt to establish its location indicated by latitude and longitude. Given a pair of such coordinates, this projectlet will compute an estimate of the distance between the locations. This can only be an estimate since we do not have any information about the elevation component of the location.
- There are many services on the web that provide an API to access information. In this project, one such service `http://api.ipstack.com/` provides the geographic coordinates of a given public node IP address (or name). It is often the case that such services return the requested information in the form of a **json** string. This leads to a need to process json formatted strings.
- Most public APIs require a request to be authenticated using an API Key which is obtained by registering with the service. In this project, we leverage the **resource** utility to encrypt this key instead of including it in clear text in the binary. Clearly the code in a public repository defeats the security but serves the purpose as an illustration.
- Utilization of the numerous utilities invoking them programmatically. In this projectlet we use the **traceroute** utility to enumerate the nodes in the path to the destination.

1.2 References

- Given the GPS coordinates, the calculation of the distance between them is done using **haversine** formula `https://www.movable-type.co.uk/scripts/latlong.html`.
- Further explanation of **haversine** is found at `https://reference.wolfram.com/language/ref/Haversine.html` and `https://en.wikipedia.org/wiki/Versine#hav`.

2 Implementation

An example in **C++** is developed using the following external libraries:

Libraries

- Accessing web services is enabled by `https://curl.haxx.se/libcurl/`.
- JSON processing is enabled by `https://curl.haxx.se/libcurl/`.
- Interacting with the system shell to execute commands is enabled by **glib** (`https://developer.gnome.org/glib/stable/`). The regular expression support of this library is used to analyze output of the commands.
- Simple encryption support from `https://gitlab.com/cpp8/bindata.git` to hide the API key in the binary.

2.1 Specification

```
./netgeo
netgeo - Version 0.1 - Sat Sep  5 10:31:12 2020
usage: netgeo <options> ipspec [ipspec...]
         -v --verbose    Verbose
         -h --help       Help
         -t --trace - with this option, a traceroute to the argument is done
                      and the distances of each hop is shown
                      without the trace option, the distances of each of the
                      command line arguments is shown
```

Implementation in **C++** : `https://gitlab.com/cpp8/netgeo.git`

2.2 Potential enhancements

- Other free services on the cloud can be queried for coordinates of Cities. Distance computation between cities might be an interesting exercise.
- The API key encryption is simple built around **xor** operations. This is the key idea in symmetric encryption algorithms the security of this coming from the protection of the key used in encryption. In this example a simplistic solution is implemented. The issue affords opportunities for experimentation.